数据科学案例分析

邪恶机器
1.01^365 = 37.8 | 0.99^365 = 0.03

1. 引言

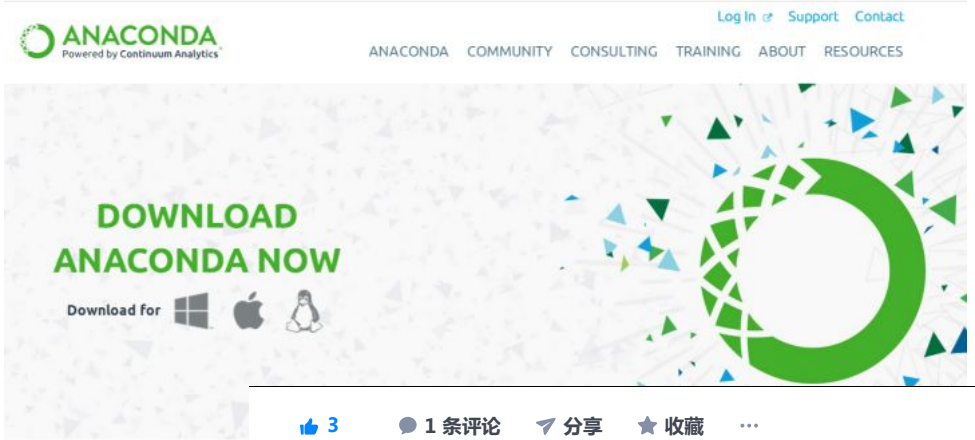
在你阅读这个帖子的第一句话时，全世界已经生成了 TB 级的数据 (1 TB = 1024 GB)。为了应付这种大规模的数据流入，数据科学 (data science) 领域在过去十年中占据了前列。 数据科学由不同领域 - 统计学，物理学，计算机科学，以及更多领域 - 融合在一起，数据科学家职业也被哈佛商业评论 (Harvard Business Review) 评为 21 世纪最性感的工作。

本贴用一个 Python 数据分析流程，展示典型的数据科学工作流程。整个流程是在 Jupyter 上完成 (Jupyter 诞生于 2014年，它支持所有编程语言的交互式数据科学和科学计算)。

【加微信公众号 MeanMachine1031，在对话框回复 DS1 可下载代码 (ipython notebook格式) 和数据 (csv格式)】

2. Python 安装

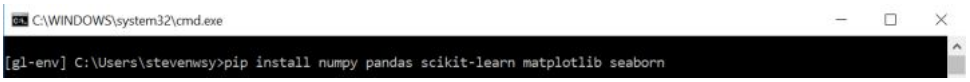
如果你的计算机上没有 Python，可以使用 Anaconda Python 发行版来安装你需要的大部分 Python 包。主页网址 (点我)，网页如下图所示：



此外，本文需要使用了几个 Python 的包是：

- numpy: 提供快速数字数组结构和辅助函数。
- pandas: 提供一个数据表结构并能高效地处理数据。
- scikit-learn: 用于机器学习。
- matplotlib: 用于基本绘图。
- seaborn: 用于高级统计绘图。

要确保你具有所需的所有软件包，在终端窗口 (terminal window) 使用 conda 或者 pip 安装它们：



3. 案例分析

假设我们要创建一个智能手机应用程序，从智能手机拍摄的照片中自动识别花的种类。 我们正在与一个数据科学家团队合作，该数据科学主管负责创建一个演示机器学习模型，测量花的萼片长度 (sepal length)，萼片宽度 (sepal width)，花瓣长度 (petal length) 和花瓣宽度 (petal width) 四个变量，并根据这些测量识别物种。

等等，萼片是什么鬼？萼片是花的最外一环。下图清晰指出花的萼片和花瓣。

我们已经从现场研究人员获得了一个数据集，里面包括三种类型的鸢尾花的测量，如下图：

根据当地研究人员测量的每种鸢尾花的四个数据 (萼片长/宽和花瓣长/宽)，我们最终目的是想正确的分类这三种花。但重中之重的第一步是数据处理。有了干净的数据用来机器学习很容易

- Python 爱好者可以用 scikit-learn
- R 爱好者可以用 Comprehensive R Archive Network (CRAN)
- Matlab 爱好者可以用 Machine Learning Toolbox (MTL)

但怎么处理数据有时候更像一门艺术而不像一门科学。在下一节我会从“回答问题”，“检查数据”，“清理数据”和“测试数据”几个方面来探索。

4. 数据处理

4.1 回答问题

任何数据分析项目的第一步就是提出想要解决的问题，并为成功解决该问题而定义一个度量 (还记得机器学习第一帖里的**性能度量**这个概念吗)。一些常见的问题如下：

- **在查看数据之前是否了解数据分析问题的类型，是回归，分类还是聚类问题？(明晰问题本质)**
  - 这是个根据萼片长度，萼片宽度，花瓣长度和花瓣宽度四个测量指标的分类问题
- **是否在一开始就定义了成功的度量？(设定量化指标)**
  - 因为是分类问题，所以可以使用查准率，即正确分类花的百分比，来量化模型的表现。我们的数据主管告诉我们应该实现 90% 的准确性。
- **现有数据是否解决分类问题？(了解数据局限性)**
  - 我们目前的数据集只有三种类型的鸢尾花。从这个数据集建立的模型将只适用于那些鸢尾花，未来创建一个通用的花分类器需要更多的数据。

请注意，我们提出这些问题时没有编写一行代码甚至查看一行数据。思考这些问题执行有效数据分析的重要一步，它往往被忽视，希望你们不要。

4.2 检查数据

即便是政府或银行，他们公布的数据也有错误。在花费太多时间分析数据之前，提早检查并修正这些错误能节省大量时间。一般来说，我们希望回答以下问题：

- 数据格式有什么问题吗？
- 数据数值有什么问题吗？
- 数据需要修复或删除吗？

接下来用 python 在 Jupyter Notebook 里展示如何检查和修正数据。

首先引进 python 里面的几个包，numpy 是为了做数学运算，pandas 是为了处理数据，matplotlib 是为了画图，seaborn 是为了画高级图。代码如下图：

```
import numpy as np
import pandas as sb
import seaborn as sb

# This line tells the notebook to show plots inside of the notebook
% matplotlib inline
import matplotlib.pyplot as plt
```

"import A as B" 这个格式是说引进 A 包用假名 B 来代替，为什么用假名呢？你看看上图里假名 B 是不是比实名 A 要简单。当你要想用 A 包里面的函数 f，你可以写成 A.f 或 B.f，但显然写成 B.f 使得代码便于读写！Python 程序员喜欢用 np, pd 和 sb 来代替 numpy, pandas 和 seaborn，当然你有自由定你的 B。假名 B 除了可以代替一个包，还可以代替一个包里的函数，比如 plt 通常代替 matplotlib 包里的 pyplot 函数。

检查点 1. 数据格式 (format)

首先用 pandas 读取 csv 文件并将数据存成数据表 (data frame) 格式。

```
iris_data = pd.read_csv('iris-data.csv', na_values=['NaN'])
iris_data.head(10)
```

	sepal_length_cm	sepal_width_cm	petal_length_cm	petal_width_cm	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	NaN	Iris-setosa
8	4.4	2.9	1.4	NaN	Iris-setosa
9	4.9	3.1	1.5	NaN	Iris-setosa

```
iris_data.tail()
```

	sepal_length_cm	sepal_width_cm	petal_length_cm	petal_width_cm	class
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	2.3	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

1. read_csv(filename, na_values = ['NaN']) 是 pandas 里面用来从 csv 文件读取数据的函数。里面用到的两个参数，第一个 filename 是读取 csv 文件名，第二个参数用来把 csv 里面空白处用 NaN 代替。此行代码将 csv 里的数据转成 pandas 里的数据表，命名为 iris_data。
2. 读取完可用 head(10) 和 tail() 函数来看前 10 个数据和后 5 个数据，如果括号里面没有指定参数，默认为 5。

从上图可知，数据似乎是存成可用的格式 (**还记得大神 Hadley Wickhan 怎么定义干净数据吗？每一列代表一个变量；每一行代表一个记录**)。

- 数据的第一行定义了列标题，标题的描述足以让我们了解每个列代表的内容 (萼片长度，萼片宽度，花瓣长度和花瓣宽度)，标题甚至给我们记录测量的单位 (cm, 厘米)
- 第一行之后的每一行代表一个花的观测数据：四个测量指标和一个类 (class)，它告诉我们花的种类。比如前 10 个都是山鸢尾花 (注意第 8 到 10 个的花瓣宽度没有数据，用 NaN 来表示)，而后 5 个都是维吉尼亚鸢尾花。

检查点 2. 数据统计 (statistics)

接下来，检查数据的分布可以识别异常值。我们从数据集的汇总统计数据开始。

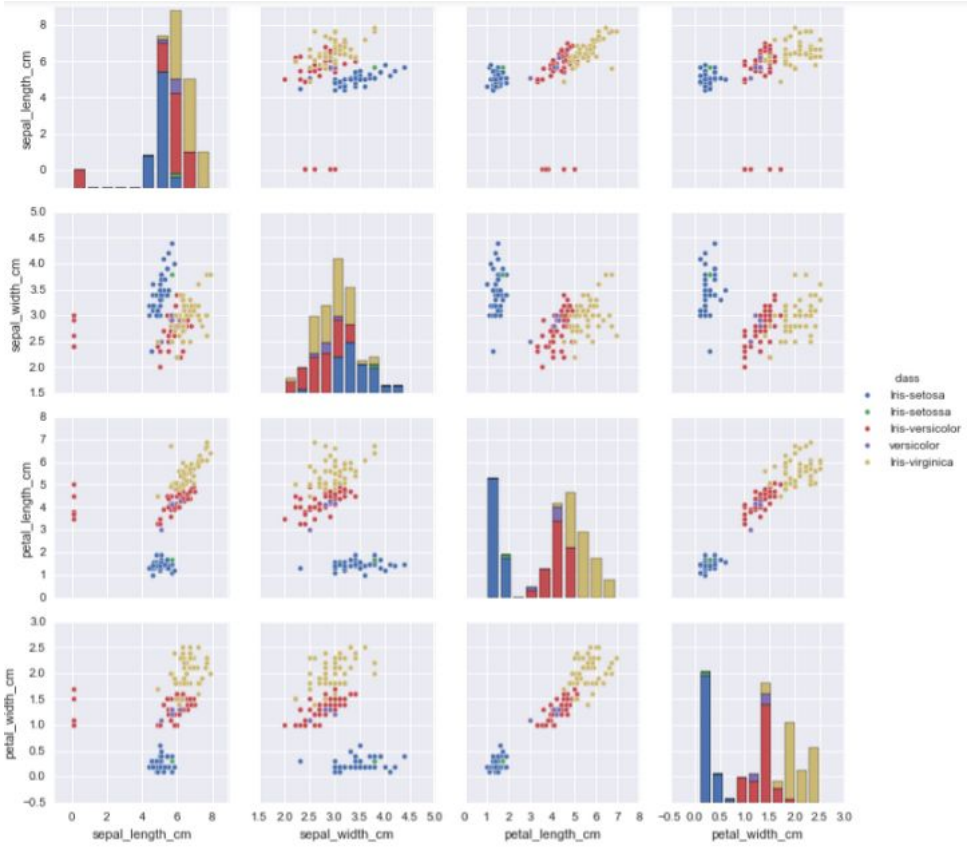
```
iris_data.describe()
```

	sepal_length_cm	sepal_width_cm	petal_length_cm	petal_width_cm
count	150.000000	150.000000	150.000000	145.000000
mean	5.644627	3.054667	3.758667	1.236552
std	1.312781	0.433123	1.764420	0.755058
min	0.055000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.400000
50%	5.700000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

1. describe() 函数的产出每列数据的个数 (count)，均值 (mean)，标准差 (std)，最小值 (min)，25, 50 和 75 百分位数 (25%, 50%, 75%) 和最大值 (max)。50 百分位数也就是中位数 (median)。
2. 从该表中看到几个有用的值。 例如，我们看到缺少 5 条花瓣宽度的数据(表里 count 那一行的萼片长度，萼片宽度和花瓣长度的个数都是 150 个，唯独花瓣宽度是 145 个)。

此外，这样的表给不了太多有用信息，除非我们知道数据应该在一个特定的范围 (如萼片长度的最小值是 0.055, 和它其他指标如均值和几个百分位数都不是一个数量级的，很有可能是测量错误)。

比起一串枯燥的数值，我更喜欢绚烂的绘图。因此我喜欢可视化数据，它能使异常值立即脱颖而出。



接下来我们用 pairplot 函数创建一个散点矩阵图，函数里

- 第一个参数 iris_data.dropna() 就是除去 NaN 的数据表，这么做原因很简单，图里不可能显示的出 NaN 值的。
- 第二个参数 hue = 'class' 就是根据类 (class) 下不同的值赋予不同的颜色 (hue 就是色彩的意思) 。

让我们再回顾一下 iris_data 的样子：

```
iris_data = pd.read_csv('iris-data.csv', na_values=['NaN'])
iris_data.head(10)
```

	sepal_length_cm	sepal_width_cm	petal_length_cm	petal_width_cm	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	NaN	Iris-setosa
8	4.4	2.9	1.4	NaN	Iris-setosa
9	4.9	3.1	1.5	NaN	Iris-setosa

它有 5 列，前四列 (萼片长度，萼片宽度，花瓣长度和花瓣宽度) 可看成自变量，第五列 (类) 可看成变量。散点矩阵图绘制前四列变量的相关系数图，而且用不同颜色区分不同的类下面的这四个变量。 从上图可知，横轴纵轴都有四个变量，那么总共可以画出 16 (4*4) 张小图。

- 对角线上的 4 张都是某个变量和自己本身的关系，由于自己和自己的相关系数永远是 1，画出相关系数图意义不大，因此这 4 张都是堆栈条形图 (stacked bar chart)。这种图是用于分解和比较整体和部分的图，说白了就是不同类都对应着一个条形图 (bar chart)，堆栈条形图就

是将所有条形图加总。

- 非对角线的 12 张就是某个变量和另一个变量的关系。比如第一行第二列的图描述的就是萼片长度 (看纵轴第一个 sepal_length_cm 字样) 和萼片宽度 (看横轴第二个 sepal_width_cm 字样)。

从散点矩阵图中，我们可以迅速看出数据集的一些问题：

1. 图的右侧标注这五个类 (Iris-setosa, Iris-setossa, Iris-versicolor, versicolor, Iris-virginica)，但原本要分类的花只有三类 (Iris-setosa, Iris-versicolor, Iris-virginica)。这意味着在记录数据时可能会犯下一些错误。
2. 在测量中有一些明显的异常值可能是错误的。

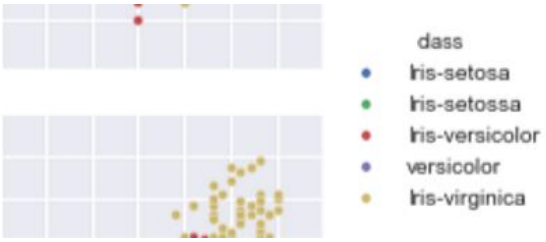
- 例如第一行后三张小图，对于 Iris-setosa (山鸢尾花，蓝点)，一个萼片宽度值落在其正常范围之外
- 例如第二行第一，三，四张小图，对于 Iris-versicolor (变色鸢尾花，红点) ，几个萼片长度值都接近零。

下一步我们的任务是要处理错误的数据。

4.3 清理数据

修正点 1. 数据类 (class)

问题：按理应该只有三个类，图中却显示五个。



在与现场研究人员交谈后，听起来像是一位研究员忘记在 Iris-versicolor 之前添加 Iris-。另一个类 Iris-setossa 他们只是多打了一个 s。让我们使用代码来修复这些错误。

```
# check the number of classes
iris_data['class'].unique()
```

```
array(['Iris-setosa', 'Iris-setossa', 'Iris-versicolor', 'versicolor',
       'Iris-virginica'], dtype=object)
```

首先用 unique() 函数 (unique 有唯一不重复的意思) 来看 iris_data 里面不重复的类名称，发现里面有五种，分别是 Iris-setosa, Iris-setossa, Iris-versicolor, versicolor 和 Iris-virginica (和散点矩阵图显示的一样)。

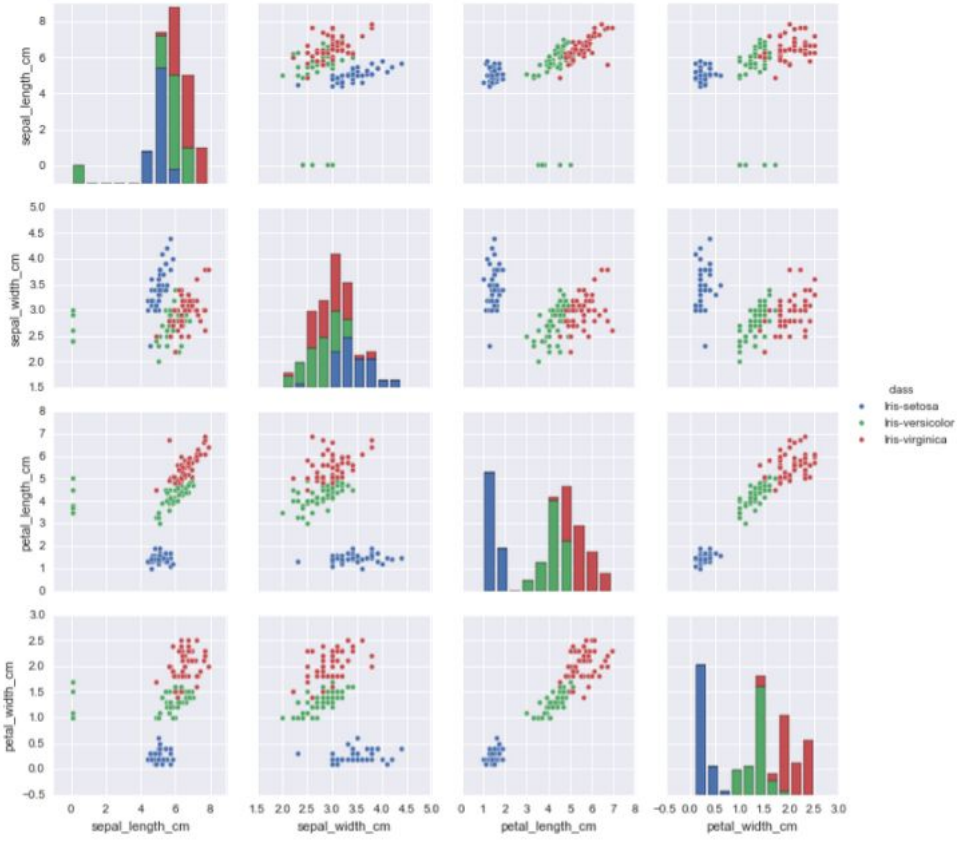
```
iris_data.loc[iris_data['class'] == 'versicolor', 'class'] = 'Iris-versicolor'
iris_data.loc[iris_data['class'] == 'Iris-setossa', 'class'] = 'Iris-setosa'

iris_data['class'].unique()
```

```
array(['Iris-setosa', 'Iris-versicolor', 'Iris-virginica'], dtype=object)
```

接着用数据表里的 loc[] 函数 (loc 是 location 的缩写，有定位之意) 来找到类值为 versicolor 的所有行，被将其类的值更新为 Iris-versicolr；同理找到类值为 Iris-setossa 的所有行，被将其类的值更新为 Iris-setosa。再用 unique() 函数检验修改过的数据是不是只有三类，更新后的散点矩阵图

如下：

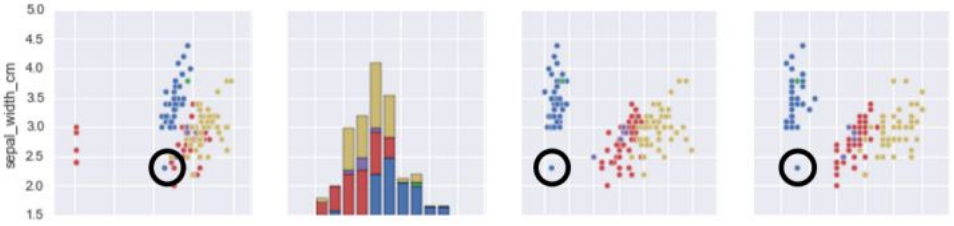


完美！只有三个类而分别是 Iris-setosa, Iris-versicolor 和 Iris-virginica。

修正点 2. 异常数值 (outliers)

修复异常值是一件棘手的事情。因为我们很难判断异常值是否由测量误差引起，或者是不正确的单位记录数据，或者是真正的异常。如果我们决定排除任何数据，需要记录排除的数据并提供排除该数据的充分理由。由上节所知，我们有两种类型的异常值。

问题 1：山鸢尾花的一个萼片宽度值落在其正常范围之外 (黑色圆框)。



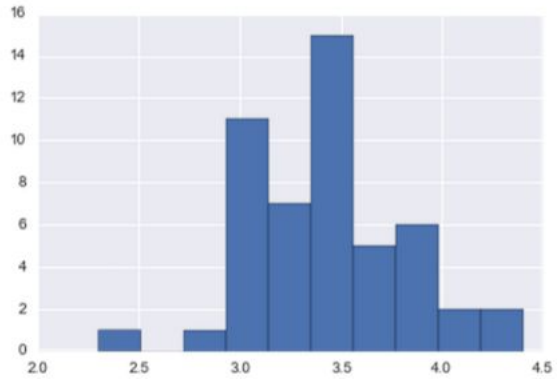
我们的研究人员知道，山鸢尾花 (Iris-setosa) 的萼片宽度 (sepal_width_cm) 不可能低于 2.5 厘米。显然，这个记录是错误的，这种情况下最有效的方法是删除它而不是花时间查找原因。但是，我们仍需要知道有多少个类似这样的错误数据，如果很少删除它没有问题，如果很多我们需要查明原因。

```
# This line finds out any 'Iris-setosa' rows with a sepal width less than 2.5 cm
iris_data.loc[(iris_data['class'] == 'Iris-setosa') _
              & (iris_data['sepal_width_cm'] <= 2.5)]
```

	sepal_length_cm	sepal_width_cm	petal_length_cm	petal_width_cm	class
41	4.5	2.3	1.3	0.3	Iris-setosa

```
iris_data.loc[iris_data['class'] == 'Iris-setosa', 'sepal_width_cm'].hist()
```

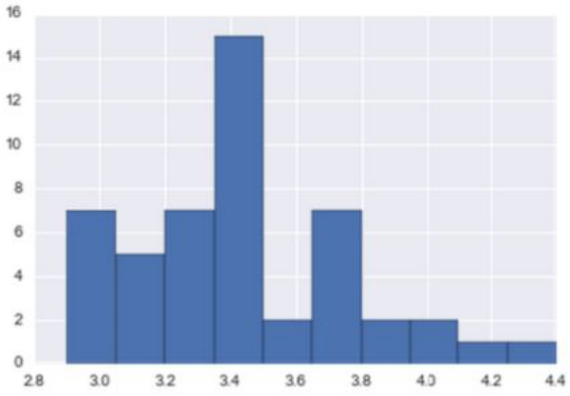
```
<matplotlib.axes._subplots.AxesSubplot at 0xefb87b8>
```



第一行代码是用数据表里的 loc[] 函数来找到类值为 Iris-setoa (因为是蓝点) 并且 sepal width 小于 2.5 的所有行。最后发现只有一个这样的数据，而上图的条形图也确认了这样的异常值只有一个。因此可以直接删除此数据。

```
# This line drops any 'Iris-setosa' rows with a separal width less than 2.5 cm
iris_data = iris_data.loc[(iris_data['class'] != 'Iris-setosa') |_
                          (iris_data['sepal_width_cm'] >= 2.5)]
iris_data.loc[iris_data['class'] == 'Iris-setosa', 'sepal_width_cm'].hist()
```

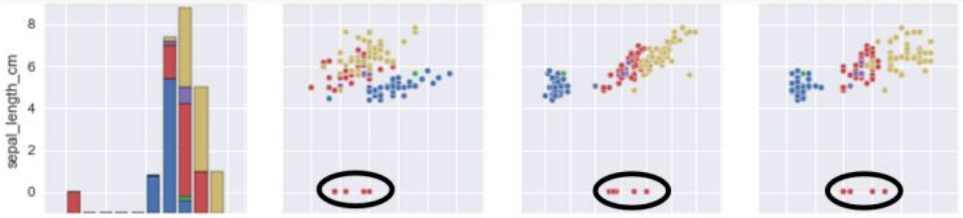
```
<matplotlib.axes._subplots.AxesSubplot at 0x104efb00>
```



第一行代码将类值不是 Iris-setosa 并且 sepal width 大于 2.5 的所有数据都新存到 iris_data 中。从上面条形图也看到了再没有这个异常值。

完美！ 现在所有的山鸢尾花的萼片宽度都大于 2.5 厘米。

问题 2：变色鸢尾花的几个萼片长度值接近与零 (黑色椭圆框)。

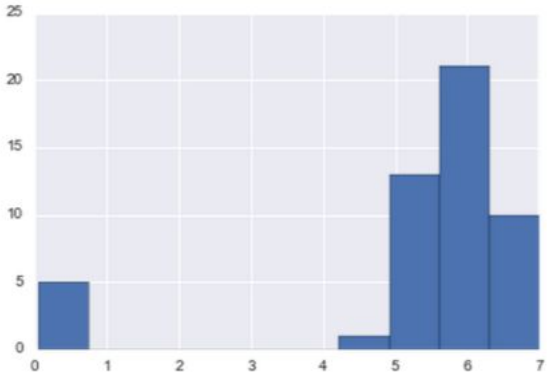


所有这些接近零的 sepal_length_cm 似乎错位了两个数量级，好像它们的记录单位米而不是厘

米。在与实地研究人员进行了一些简短的对话后，我们发现其中一个人忘记将这些测量值转换为厘米。让我们使用代码来修复这些错误。

```
iris_data.loc[(iris_data['class'] == 'Iris-versicolor'),'sepal_length_cm'].hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x10909898>
```

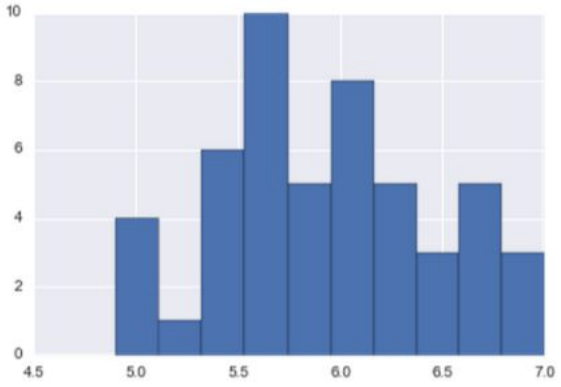


第一行代码是用数据表里的 loc[] 函数来找到类值为 Iris-versicolor (因为是红点) 并且 sepal length 接近零的所有行，发现有五个数据，而条形图最左边显示的数据个数也确认了是五个。

```
iris_data.loc[(iris_data['class'] == 'Iris-versicolor') &
              (iris_data['sepal_length_cm'] < 1.0),
              'sepal_length_cm'] *= 100.0

iris_data.loc[iris_data['class'] == 'Iris-versicolor', 'sepal_length_cm'].hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0xc2a8e48>
```



第一行代码将类值为 Iris-versicolor 并且 sepal length 都小于 1 的所有数据乘以 100 (将米转成厘米)。从上面条形图也看到了再没有这五个异常值。

完美！ 我们成功修正了这些异常值，要不然我们的分析结果可能毫无意义。

修正点 3. 缺失数值 (missing value)

对了，我们还有些 NaN 数据。通常我们有两种方式来处理这类数据。

1. 删除 (deletion)
2. 插补 (imputation)

在本例中删除不是理想的做法，特别是考虑到它们都在 Iris-setosa 下，如图：

A	B	C	D	E
sepal_length_cm	sepal_width_cm	petal_length_cm	petal_width_cm	class
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5	3.4	1.5	NaN	Iris-setosa
4.4	2.9	1.4	NaN	Iris-setosa
4.9	3.1	1.5	NaN	Iris-setosa
5.4	3.7	1.5	NaN	Iris-setosa
4.8	3.4	1.6	NaN	Iris-setosa

所有缺失的值都属于 Iris-setosa类，直接删除可能会对日后数据分析带来偏差。此外，可以用插补方法，其最常见的方法平均插补 (mean imputation)。其做法就是“假设知道测量的值落在一定范围内，就可以用该测量的平均值填充空值”。

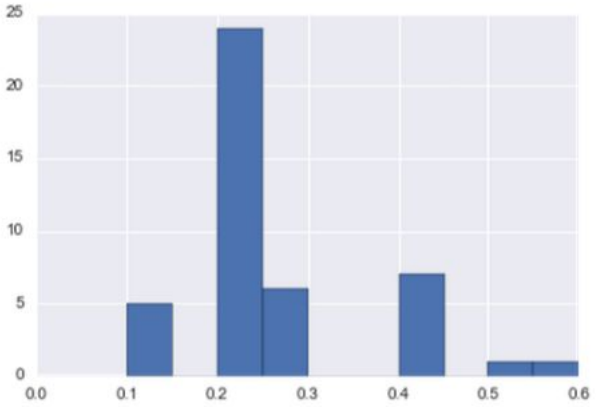
```
iris_data.loc[(iris_data['sepal_length_cm'].isnull()) |
              (iris_data['sepal_width_cm'].isnull()) |
              (iris_data['petal_length_cm'].isnull()) |
              (iris_data['petal_width_cm'].isnull())]
```

	sepal_length_cm	sepal_width_cm	petal_length_cm	petal_width_cm	class
7	5.0	3.4	1.5	NaN	Iris-setosa
8	4.4	2.9	1.4	NaN	Iris-setosa
9	4.9	3.1	1.5	NaN	Iris-setosa
10	5.4	3.7	1.5	NaN	Iris-setosa
11	4.8	3.4	1.6	NaN	Iris-setosa

上面代码里面 iris_data[A].isnull() 语句是找出 A列中值为 "NA", "NaN" 的行数据，而 "|" 是“或”的意思，因此上面整句话是找到萼片长度，萼片宽度，花瓣长度和花瓣宽度这四列下的所有含 NaN 的行数据，最后发现只有 5 个，而且 NaN 都来自花瓣宽度。

```
iris_data.loc[iris_data['class'] == 'Iris-setosa', 'petal_width_cm'].hist()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x108dd898>
```



接下来用 hist() 函数画出 Iris-setosa 花瓣宽度的条形图，可以清楚看到大多数宽度在 0.25 左右。

```
average_petal_width = iris_data.loc[iris_data['class'] == 'Iris-setosa',
                                    'petal_width_cm'].mean()

iris_data.loc[(iris_data['class'] == 'Iris-setosa') &
              (iris_data['petal_width_cm'].isnull()),
              'petal_width_cm'] = average_petal_width

iris_data.loc[(iris_data['class'] == 'Iris-setosa') &
              (iris_data['petal_width_cm'] == average_petal_width)]
```

	sepal_length_cm	sepal_width_cm	petal_length_cm	petal_width_cm	class
7	5.0	3.4	1.5	0.25	Iris-setosa
8	4.4	2.9	1.4	0.25	Iris-setosa
9	4.9	3.1	1.5	0.25	Iris-setosa
10	5.4	3.7	1.5	0.25	Iris-setosa
11	4.8	3.4	1.6	0.25	Iris-setosa

然后用 mean() 准确求出其宽度的平均值，将其 NaN 值全部用平均值代替，最后打出那 5 行插补后的数据表。

```
iris_data.loc[(iris_data['sepal_length_cm'].isnull()) |
              (iris_data['sepal_width_cm'].isnull()) |
              (iris_data['petal_length_cm'].isnull()) |
              (iris_data['petal_width_cm'].isnull())]
```

	sepal_length_cm	sepal_width_cm	petal_length_cm	petal_width_cm	class

为了确保所有 NaN 值已被替换，再次用 iris_data[A].isnull() 语句来查看，出来的结果是一个只有列标题的空数据表。这表示表内已经没有 NaN 值了。

完美！完美！完美！

4.4 测试数据

存储数据 (save data)

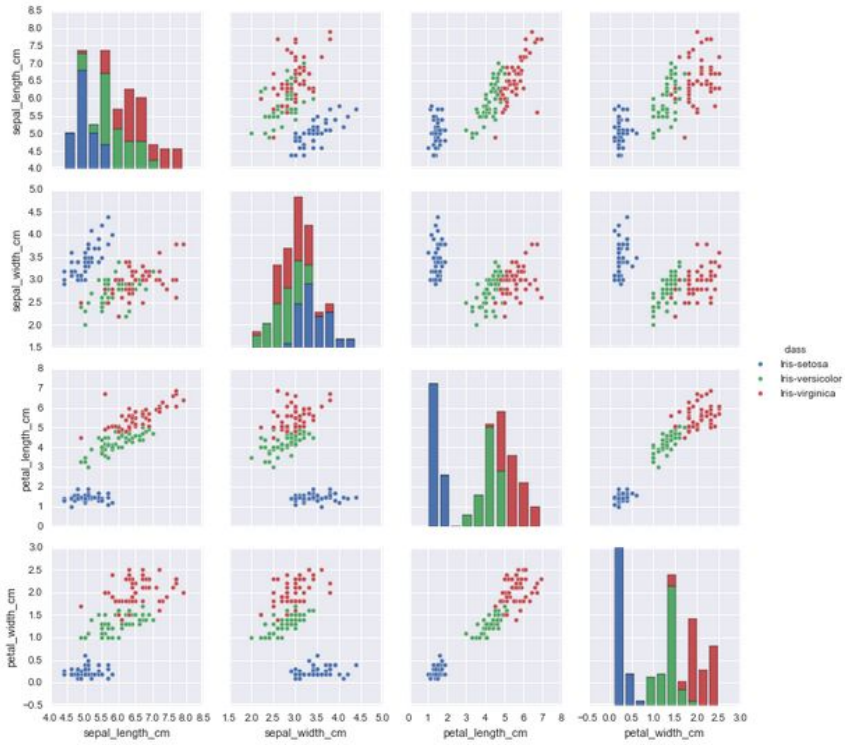
经过所有这些艰苦的工作，我们不想在每次使用数据集时都重复这个过程。让我们将整洁的数据文件保存为一个新文件，日后做数据分析可以直接从该文件开始。

```
iris_data.to_csv('iris-data-clean.csv', index=False)

iris_data_clean = pd.read_csv('iris-data-clean.csv')
```

第一行代码 to_csv() 函数是把整理后得数据写道一个名叫 iris-data-clean 的 csv 文件里，index = False 意思是不要记录在csv 里记录行数。第二行代码重新读取刚写好的 csv 并存成名为 iris_data_clean 的数据表。

让我们再看看基于干净数据画的散点矩阵图吧



从上图可看到：

- 五个类变成三个类。
- 异常值全部删除或修正了。

整个图看起来整洁多了。

声明数据 (assert data)

为了防止一些数据问题没有解决，我们可以用 assert 语句来做声明。该语句好处是，在运行时如果声明语句为真，没有任何事发生，反之会报错而警告我们有哪些错误数据需要注意且修正。

```
# We know that we should only have three classes
assert len(iris_data_clean['class'].unique()) == 3
```

```
# We know that sepal lengths for 'Iris-versicolor' should never be below 2.5 cm
assert iris_data_clean.loc[iris_data_clean['class'] == 'Iris-versicolor',
                           'sepal_length_cm'].min() >= 2.5
```

```
# We know that our data set should have no missing measurements
assert len(iris_data_clean.loc[(iris_data_clean['sepal_length_cm'].isnull()) |
                               (iris_data_clean['sepal_width_cm'].isnull()) |
                               (iris_data_clean['petal_length_cm'].isnull()) |
                               (iris_data_clean['petal_width_cm'].isnull())]) == 0
```

上述三个声明分别是：

1. 花应该只有三个类型。
2. 变色鸢尾花的萼片长度应该大于 2.5 厘米。
3. 数据不应该有缺失。

如果任何声明被违反，我们应该立即停止分析，而回到整理阶段。

5. 小结

小贴士：

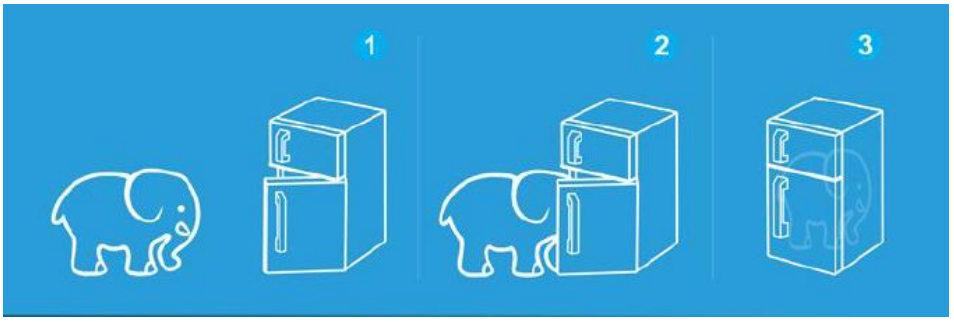
1. **确保数据在预期范围内，并尽可能使常识来定义预期范围处理丢失的数据**
2. **根据情况删除它或者更换它**
3. **永远不要手动清理数据，数据量一大你会发现你根本无法复制此过程**
4. **使用代码来记录如何整理数据的过程，Jupyter Notebook 真的是个很好的工具**
5. **将数据可视化，和字符相比，图永远更直观，也是更容易发现数据问题**

6. 彩蛋

电影都有彩蛋，帖子为什么不能有？本彩蛋只想证实有了干净数据，用 scikit-learn 来机器学习不要太容易。不信？下图文飨你！

要把大象放冰箱总共分几步？三步！

1. 把冰箱门打开
2. 把大象放进去
3. 把冰箱门关上



完了要掉粉了，言归正传。碰巧的是有了干净数据后，机器学习也分三步：

1. 选出特征 (输入变量) 和标记 (输出变量)

```
# Read data
iris_data_clean = pd.read_csv('iris-data-clean.csv')

# Assign features variable
all_inputs = iris_data_clean[['sepal_length_cm', 'sepal_width_cm',
                              'petal_length_cm', 'petal_width_cm']].values

# Assign label variable
all_classes = iris_data_clean['class'].values
```

all_input 是特征，all_classes 是标记。有标记那么此分类问题是监督学习。

2. 划分训练集和测试集

```
(training_inputs, test_inputs, training_classes,
 test_classes) = train_test_split(all_inputs, all_classes, train_size=0.75, random_state=1)
```

用 75% 和 25% 的比例来划分训练集和测试集，设一个 random_state 是为了在机器学习中每次出的结果一样，便于找问题。

3. 用模型来学习

```
# Create the classifier
decision_tree_classifier = DecisionTreeClassifier()

# Train the classifier on the training set
decision_tree_classifier.fit(training_inputs, training_classes)

# Validate the classifier on the testing set using classification accuracy
decision_tree_classifier.score(test_inputs, test_classes)
```

```
0.97368421052631582
```

再一次感叹 scikit-learn 的强大和好用，三行代码解决分类问题了。第一行创建一个决策树分类器变量。第二行用 fit() 函数在训练集上学习。第三行用 score() 在测试集上评估。

噢耶！好像没费什么力气，我们的查准率就达到了 **97%**！远高于主管给我们的 90% 要求。

这时候有些机器学习专家会说了：

- 这个训练集和测试集划分是随机的，一次不足以说明你查准率高。
- 这数据太少没代表性，换套数据模型可能过拟合。
- 特征选择有很大学问，凭什么就选萼片长度，萼片宽度，花瓣长度和花瓣宽度这四个变量？

是的，你说的这些都是 legit 的，但是本文核心是想说数据清理后机器学习会更容易，我只是举个例子。再者，我此贴把上面的问题全解决了后面的帖子怎么出？跟着我继续在数据科学和机器学习中继续探索咯！Stay Tuned!

编辑于 2016-12-01

数据可视化　数据科学家

推荐阅读

数据科学完整学习路径——R语言 附书籍下载

数据小咖　发表于数据小咖的...

如何成为一名数据科学家 | 学习篇(附视频中字)

Data ...　发表于Data ...

数据科学家的离职潮背后，有怎样的扎心真相？

大腿君　发表于DT财经

打造 习项

EarlG

1 条评论　⇌ 切换为时间排序

写下你的评论...

张幼薇　1 年前

非常详细，最近在看seaborn 讲得很透彻，谢谢！

赞